Abstract
We reflect on the current state of sketch based interaction and offer sketchy thoughts on where interesting work can be done.

Keywords
sketch, diagram, interaction, intelligent paper

ACM Classification Keywords
H5.m. Information interfaces and presentation: Miscellaneous.

General Terms
pen based interaction

Introduction
It’s been almost fifty years since the first sketch-based computing efforts, and depending on how you look at it we’ve come a long way. The hardware: from Sutherland’s light pen drawing on a vector display[8], to the paper and ink Anoto system and now e-paper displays. The software too, from crude feature detection and simple recognizers to increasingly subtle, sophisticated, and versatile ones. Although certainly we have not solved all the technical problems, as a community each year we make advances that bring us closer to building truly useful sketch-based computing systems. Let’s give ourselves the benefit of the doubt, and assume that over the next N years (you pick the N) we’ll have reliable enough recognizers, responsive and high enough resolution digitizing displays, and
interaction techniques that are appropriate for pen based applications.

What will we want to do with them? We might have asked this question more forcefully before, because one common critique of research in sketch based interaction and modeling (SBIM) is that we have no “killer” application. For a while, with the Apple Newton and then the Palm Pilot, it looked like the personal digital assistants (PDAs) would be pen-based. But then PDAs were replaced by the cell phone and its keypad. And although many thought that people would use pens to write, even with a stylized alphabet like Graffiti, it turns out that people are content to use a physical or soft keyboard to compose text.

Drawing plays a strange role in our lives. On one hand, like gesture and speech, we can all draw a diagram when we need to. On the other hand, except for those of us who deliberately learn to draw as part of our disciplinary training, most of us don’t consider ourselves “able to draw”. Many people express that they “wish they could draw”, or when asked to draw something, say “I can’t draw.” Many see drawing as either an artistic, ‘creative’ skill or else a technical one that requires extensive training.

In our earlier work, we argued that drawing (diagramming and sketching) are not so much the hallmark of creativity (though of course, many creative people draw, and drawing can support creativity), but that drawing is a means to reason and think [3]. We called our Electronic Cocktail Napkin project, “a freehand drawing interface for knowledge based systems” [4]. We deliberately took a meta-disciplinary approach because we felt strongly that drawing isn’t just for architects, or artists, or engineers. We wanted to show that the tools for building a system for one domain could be used for building a system for another. We were interested in drawing as a tool for thinking, in using drawing to think about thinking, and in building systems and tools to support this form of representation and reasoning.

In this we diverge from those researchers who are primarily interested in using drawing to produce visual representations, for example the “M” in the SBIM (Sketch Based Interaction and Modeling) community. To be sure, there is a rich body of work on pen based systems and tools for making three-dimensional models from sketches, and this has tremendous application in the computer graphics industry—it is a way to provide an interface for artists and modelers to use the familiar pen to do their work. We admire this research; we’ve even done a little of this ourselves[2, 7]; but in the end our main interest is in drawing as thinking.

If sketch based interfaces are to be more than a novelty, and useful to more than artists, then we (the sketching research community) should understand why we draw[6], and how drawing helps us think [9]. This should lead to a better understanding of the space of potential applications, and perhaps in time, to the “killer” application that would make sketch based computing take off. (Or perhaps not.)

We are aware, of course, of a wide variety of prototypes and demonstration systems that the sketching research community has built, and in these may well be the seeds of more mature applications.
Why we draw

Drawing to explain
If you found myself in a foreign land where you didn’t speak the local language, and needed to buy some essential item, you might resort to drawing a little picture to explain what you want. We’ve even done this in a hardware store at home, when we need some doohickey for a plumbing repair job, but we don’t know its name. Usually making a simple sketch is enough for the sales clerk to help me find the needed item. This kind of sketch needn’t be artistic. It need only capture the features of the thing that we both recognize as essential and defining. Quite often these conversations proceed through two or more successive stages. For example, you walk in and say, “I’m looking for a washer to fix a dripping faucet, it looks like this.” and you draw a little diagram:

But the sales clerk, asks, “Is it flat or round?”

This question doesn’t make sense (obviously, it’s round: you just drew a circle!). “Huh?” you say, “I don’t understand.”

The clerk says, “if you slice through it, would it look flat or round?” and draws this cut-section sketch:

And now we understand the difference between a flat washer and an O-ring.

This is a simple and somewhat contrived example; obviously one could explain the difference between a flat washer and an O-ring without drawing a picture. But the drawing helps. And it’s not hard to find more complicated examples where a verbal explanation is unwieldy and a drawing communicates clearly and succinctly.

You might also use a drawing to explain how to find something—as a map. Even in this day of GPS and Google maps, it’s still sometimes helpful to use a sketch map to navigate. For example, if you’ve agreed to go to the supermarket to buy some baby nappies, but you don’t know the store layout, we might show you where to find them by drawing a little map. Or, we might draw a sketch map to show you where to park your car when you come to visit us.

These examples are both intrinsically spatial, but drawings also help explain things that are also not spatial. Think of examples from mathematics: A Gaussian (bell-curve) distribution. Exponential growth. We draw these graph diagrams to illustrate a relationship: supply vs. demand, population growth over time. And we label them, to associate a spatial
dimension with the dimension in the target domain: “the horizontal axis represents time.”

**Drawing to propose**

If we are moving into a new apartment, we might begin by drawing a floor plan of the rooms, and then propose different arrangements of furniture by quickly sketching diagrams to scale. Implicit in this is a question: How should we arrange our furniture?, or even: Will our furniture fit? The result of this sketching activity is usually not a single sketch drawing, but a set or series of alternative drawings. The drawing is a means of virtual experiment—it’s easier to make a drawing with the refrigerator in different places than to push the fridge around the kitchen. And evaluation is an important step—we don’t just make the drawing to enjoy looking at it; we make the drawing to test whether the furniture arrangement will work. Although we don’t necessarily state the criteria explicitly (sometimes we do), we evaluate the furniture layouts we make by asking questions such as, “can I easily walk through the room?” “Is there a convenient work triangle between the stove, sink, and refrigerator” and “Does this arrangement make a cozy area to talk?”

**Drawing to solve problems**

In a similar vein to solving furniture layout problems, but somewhat more formally, experts in many domains adopt pictorial representations for stating and solving problems. Chemists use molecular diagrams to represent chemical reactions; mechanical engineers use free-body diagrams to represent static forces; electrical engineers use diagrams to represent circuits. In each of these (and other) domains, the diagrams and a set of agreed-on transformations are used to represent and reason about a domain. For example, a chemist can tell from a molecular diagram whether a chemical reaction will take place: an organic synthesis is represented by a sequence of transformations, similar to a sequence of steps in algebra or a geometric proof. Here again, the sketch isn’t just an artistic rendering, but an instrument of thought—it’s bound to quite specific rules of its construction and transformation. Cheng called drawings used this way *law-encoding* diagrams [1].

**Drawing to make**

We also draw when we prepare to make something, to consider the materials we plan to use and the methods of manufacturing them. If we’re making a shirt, we draw the shapes of the individual pieces of cloth (sleeves, collar, etc.) and consider how we will sew them together. If we’re making a physical device, we draw the parts and think about how we will assemble them. How will we hold this spring in place while we insert the plunger? What will keep the wheel from wobbling? How will the vessel remain vertical so the liquid doesn’t spill? In these cases, we apply to the diagram—in our mind—the physics that will bear on the things we make, once the thing exists in real life. This is really design—the ability to anticipate how a thing proposed will behave, based on a sketch. And in many cases it’s not limited to the physics of the thing. The designer of a building must consider how people will use the building—where they will (and won’t) walk, how the lines-of-sight will affect people’s feelings in the building, where trash will tend to accumulate, and so on.

Drawing-to-make entails more than an application like SketchUp or any of the sketching systems that help a user produce a three-dimensional model. That is
because making something requires more than specifying its three-dimensional form. It also entails a process of considering and testing, which in turn entails a great deal of domain knowledge.

So What?
All this territory is surely familiar to anyone who has worked on sketching systems in the past decade or so. Yet it is especially worth bearing in mind as we move forward from barely-working proof-of-concept systems to more robust recognition, more-than-almost-acceptable hardware, and interaction techniques that are tuned, not for the mouse, but for the pen. As we advance beyond the basic challenges that we’ve been working on as a community for the past several decades (!) we can begin to engage with some other, perhaps deeper, challenges. Many of us have tried to address these challenges, piecemeal, here and there, in our work. But it's been difficult to do this in the face of inadequate hardware, flawed or buggy recognition, and impoverished interaction techniques. And to be sure, those problems are yet unsolved. Yet perhaps it’s time to look beyond the immediate interaction challenges and think (once again, and more carefully) about the roles that drawing and visual reasoning play in thinking and how computing can support, enhance, and advance those roles.

For example, a classic and still unsolved problem in sketching research is interpreting a 2D sketch to a 3D model. An application like Google's SketchUp [http://sketchup.google.com/] arguably addresses this problem—not by solving the sketch-recognition and interpretation problem, but by providing a relatively simple interface to make 3D models. Some say, "who needs sketch recognition? – SketchUp solves the problem!" SketchUp is clearly a successful way to get users who are untrained in drawing to create 3D content, and arguably a sweet graphical user interface to modeling software. However, it sidesteps the problems we are interested in — both the technical problem of how to interpret a 2D sketch of an object or a scene, and the corollary “meaning” problem of how sketches convey information. SketchUp is a clever work-around to this problem, and it does its job well. It just doesn’t do the job we’re really interested in.

SketchUp proponents might argue that if the goal is to create 3D models then there’s no need to follow old-fashioned conventions of drawing with pen and pencil on paper; that it’s more economical and sensible to invent new ways to represent 3D data, using the interaction methods that computing affords. In the abstract we’re sympathetic to that argument, but in fact, the methods and means of representation that people have developed over centuries are still capable of far more subtlety.

Think Deeper and Better
Perhaps then, instead of speeding up, automating, or beautifying drawings, constructing a rectified drawing, or creating 3D data, we should look instead into how sketching can stretch time, or our imaginations to help us think better and deeper.

Consider an analogy with tools to support writing. Handwriting recognition per se or having more fonts in a word processor does not make for a better writing process. On the contrary, the minor recognition errors and different font sizes and faces actually distract the writer and the reader. No doubt spelling and grammar checkers as well as appropriate style formatting (Bold,
Italic, heading, etc) help authors organize the parts of a paper, but it is tools such as readability analysis* that can help an author at a more fundamental level. This statistical analysis can show you the character of your writing – the degree to which you’ve used the passive voice, the length of your sentences, and the grade level readability, etc. This doesn’t fix problems for you, but makes you aware of the context and situation, so that you can think and find ways to fix it yourself. It doesn’t require the software to understand the content of your writing, yet it addresses more than the superficial attributes of spelling, grammar, and layout.

So it is with sketching. Although eventually it’s useful to “beautify” sketches into rectified and regular diagrams for presentation [5], the main reasons we sketch are internal, for thinking, reasoning, and reflecting, and for these purposes beautification may actually be counterproductive. Likewise, although it’s certainly useful for a program to recognize the elements of a drawing when it can, it’s not always possible to do so—and at least in part because the drawing may be ambiguous even to its author. In such a case it’s better not to present incorrect or inappropriate recognition results.

As we move forward from addressing the low-level software and hardware challenges that have rightfully consumed research in sketch based interaction over the past several decades, it is time to attend to some broader questions of how sketching (drawing, diagramming) is embedded into our personal, social, and cultural practices of computing.

* Word provides readability analysis under its Tools / Grammar and Spelling menu.

About the Authors

Mark D Gross began investigating pen based interaction in 1993 out of desperation, having failed entirely to get designers interested in his earlier work on constraint based programming as a platform for building computer aided design environments. He thought that perhaps he could seduce designers to program design constraints if they were given familiar tools. He began by implementing a simple version of the Ledeen recognizer; then added some additional template features; then spatial relations; then a parser for visual language, and simulating tracing paper, and pretty soon the Electronic Cocktail Napkin project had captured him. Then he met Ellen Yi-Luen Do, who (unlike him) actually knows what to do with a pencil. Together they implemented an interface to the case-based design tool, Archie, developed by Janet Kolodner’s group at Georgia Tech, using diagrams to index and retrieve cases in a library of architectural designs. Thus began a long and fruitful collaboration exploring various aspects of sketch recognition, and pen-based interaction, support for ‘intelligent paper’, and even ventured into thinking about visual thinking.

Prof. Ellen Yi-Luen Do started drawing in early childhood. Putting pencil marks on paper (and the wall !) helped her understand the world, conquer the fear of the darkness of the long corridor, develop interesting stories, learn to solve geometry and math problems, and more. In her academic life she has worked on a variety of projects that explore freehand drawing as an interface to knowledge based design systems, including Archie’s Napkin,
Electronic Cocktail Napkin, Drawing Analogies, Shape-Based Reminding, Thinking with Diagrams, The Right Tool at the Right Time, Redliner, Space Pen, Light Pen, Light Sketch, Drawing on the Back of an Envelope, VR Sketchpad, Critiquing Freehand Sketches, etc., many with her (first mentor then) colleague Prof. Mark D Gross, and students from University of Washington, Carnegie Mellon, and now Georgia Institute of Technology. Her involvement transformed the Electronic Cocktail Napkin project from basic functions to include applications of sketchbook indexing and retrieval, visual simulation, and 3D geometry and virtual reality.

References


